REMARKS

Claims 1-24 are presented for further examination. Claims 1, 5, 10, 11, 15, 20, and 22 have been amended.

In the Office Action mailed August 31, 2005, the Examiner rejected claims 1-11 and 15-21 under 35 U.S.C. § 103(a) as being unpatentable over U.S. Patent Application Publication No. 2005/0015775 ("Russell") in view of U.S. Patent No. 6,425,017 ("Dievendorff"). Claims 14 and 24 were rejected under 35 U.S.C. § 103(a) as unpatentable over Russell and Dievendorff in view of "Official Notice." Claims 12 and 22 were rejected under 35 U.S.C. § 103(a) as unpatentable over Russell and Dievendorff in view of U.S. Patent No. 6,182,068 ("Lomet"). Claims 13 and 23 were rejected under 35 U.S.C. § 103(a) as unpatentable over Russell and Dievendorff in view of U.S. Patent No. 6,223,205 ("Harchol-Balter").

Applicants respectfully disagree with the bases for the rejections and request reconsideration and further examination of the claims.

Applicants' claims recite several aspects nowhere taught, suggested, or motivated by Russell, Dievendorff or Lomet, alone or in combination. Specifically, independent claim 1, as amended, recites inter alia, "*implementing an interfacing method internal to a cable modem*, comprising: providing the *cable modem* with server components and configuring the server components to implement a set of functions; ..." (Emphasis added.) Furthermore, claim 22, as amended, recites, "The system of claim 21, wherein the network station is a *cable modem*, the station manager is a *cable modem manager*, and the interface manager is an *OSSI client manager*." (Emphasis added.) The context of the inventions of Russell, Dievendorff, and Lomet is non-analogous to that of Applicants' invention. Both Russell and Dievendorff are directed generally to client-server transaction processing systems built upon the Component Object Model ("COM") as provided by multiple, networked workstations running Microsoft Windows operating systems. Microsoft Windows operating system(s) operate on general-purpose computing systems with substantial processing power and significant memory capacities. Lomet is directed generally to the creation of log files of client-server requests to facilitate application and database recovery.

The Applicants' claims, on the other hand, recite operation in the context of a single cable modem device. Furthermore, the systems of Russell, Dievendorff, and Lomet would be rendered inoperative if applied to the embedded processor context of cable modem devices that are operative with minimal memory and processing facilities. The cited references all require substantial operating systems-level infrastructure (e.g. COM) in addition to hardware support such as hard drives (e.g. for the storing of log files on hard disk as in Lomet) that was unavailable in the context of cable modem devices at the time of Applicants' invention.

For at least these reasons, Applicants respectfully submit that claims 1 and 22, as well as all claims depending therefrom, are clearly allowable.

In addition, Applicants' claims recite an initialization process nowhere taught, suggested, or motivated by Russell or Dievendorff, alone or in combination. Specifically, claim 1, as amended, recites, "... an interface manager, *created by the client component during the initialization of the cable modem*, ..." Claim 5, as amended, recites, "... an interface manager, *created by the second component during the initialization of the network device*, ..." Claim 10, as amended, recites, "... an interface manager, *created by the station manager during initialization of the network station*, ..." Claim 15, as amended, recites, "... an interface manager, *created by the client component during the initialization of the network device*, ..." Claim 20, as amended, recites, "... an interface manager, *created by the station manager during initialization of the network station*, ..." (Emphasis added.)

Russell does not teach, suggest, or motivate the initialization process recited by Applicants' claims. Specifically, Russell does not disclose the initialization of an interface manager by the system components (e.g. client component, station manager, second component) recited by Applicants' claims. In a cited passage, the Examiner draws a correspondence between Applicants' interface manager and Russell's transaction server executive. (see column 9, paragraph 91) (hereinafter in col#:paragraph# format). The cited passage also states that the transaction server executive loads and instantiates the server application component. It further discloses that the transaction server executive "manages calls to the server application component from client programs." It is clear from the cited passage that the transaction server executive in Russell must exist prior to the initialization of other system components (server and

client components) and therefore teaches a component creation and initialization order that is opposite from that recited in Applicants' claims. Moreover, the system disclosed by Russell would likely be rendered inoperative if the order of component creation were varied because there would be no component to "manage" client calls to server components. Russell therefore teaches away from the order of component creation recited by Applicants' claims.

Nor does Dievendorff teach, suggest, or motivate the initialization process recited by Applicants' claims. Specifically, Dievendorff nowhere describes the creation of an interface manager by a system component at the time of system initialization as recited by Applicants' claims.

For at least these reasons, Applicants respectfully submit that independent claims 1, 5, 10, 15, and 20, as well as all claims depending therefrom, are clearly allowable.

Applicants' claims also recite a message-passing architecture nowhere taught, suggested, or motivated by Russell or Dievendorff, alone or in combination. Specifically, claim 1, as amended, recites, "each request includes *a component identifier, an interface identifier, and a function parameter*; processing the client component requests by invoking the interface method corresponding to the *interface identifier* of the server component corresponding to the *component identifier* included in the request and passing the function parameter to the server component; and providing a response message regarding the execution of the interface method to the client component via a second message queue, wherein *the response message comprises a status.*" Claims 5 and 11, as amended, recite a similar message-passing architecture.

Russell does not teach, suggest, or motivate requests that include a component identifier, interface identifier, and a function parameter. In a cited passage, Russell discloses the use of an API function called "CoCreateinstance" by client programs to create new components. This function takes two parameters, a CLSID (class identifier) and an IID (interface identifier) (see Russell at 16:145). First, the CoCreateInstance function requires two parameters, while Applicants' claims recite three parameters. Second, the parameters of Applicants' messages refer to different entities than those referred to by the parameters of the CoCreateInstance function. Specifically, Russell's first parameter is a *class identifier*, whereas Applicants' first parameter is a *component identifier*. The term "class" has a specific meaning to one skilled in

the art of object-oriented programming which is significantly different from that of "component." A class is a *description* of the behavior of a set of objects that are instances of that class, whereas a component is an actual computational entity with state and behaviors.

Furthermore, Russell does not teach, suggest, or motivate the interface method lookup aspect recited by Applicants' claims. The Examiner asserts that Russell teaches this aspect at 16:145-146 and 16:148-17:149. Both passages generally describe the process of object creation by use of the CoCreateInstance method. Again, the CoCreateInstance method is used to create new objects, not to invoke functionality of server components. Moreover, the cited passages nowhere describe invoking interface methods that correspond to interface identifiers of server components corresponding to component identifiers and passing function parameters to the server components.

Finally, Russell does not teach, suggest, or motivate the provision of a response message comprising status information. The Examiner asserts that Russell teaches this aspect at 28:239 and Table 2. The cited passages describe the operation and return codes of a function called "SetComplete." The cited Table 2 may describe return values, but they relate to the operation of the SetComplete function, not the CoCreateInstance function. Previously, the Examiner drew a correspondence between the operation of Russell's CoCreateInstance function and Applicants' processing of requests by invoking corresponding interface methods. Here, the Examiner draws a correspondence between Russell's SetComplete function and Applicants' provision of response messages comprising status codes. Applicants' claims clearly recite that the response messages relate to "the execution of the interface method" described above. There is nothing in Russell to suggest that the functions CoCreateInstance and SetComplete can be made to simultaneously correspond to one of Applicants' interface methods. In fact, Russell clearly states that CoCreateInstance does not return a status, but rather returns a pointer to an interface (see Russell at 16:145). This indicates that Russell's teachings related to the CoCreateInstance method and the SetComplete method cannot be combined because they return entirely different return values.

Dievendorff does not teach, suggest, or motivate the particular message passing architecture recited in Applicants' claims. Dievendorff describes numerous COM API calls but

does not disclose a uniform message format, dispatch functionality, and does not return values as recited by Applicant's claims.

For at least these reasons, Applicants respectfully submit that claims 1, 5, and 11 as well as all claims depending therefrom, are clearly allowable.

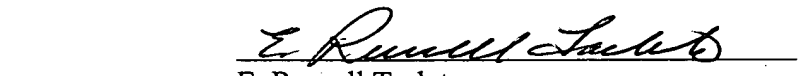
In view of the foregoing, Applicants submit that all of the claims in this application are clearly allowable. In the event the Examiner finds informalities that can be resolved by telephone conference, the Examiner is urged to contact applicants' undersigned representative by telephone at (206) 622-4900 in order to expeditiously resolve prosecution of this application. Consequently, early and favorable action allowing these claims and passing this case to issuance is respectfully solicited.

The Director is authorized to charge any additional fees due by way of this Amendment, or credit any overpayment, to our Deposit Account No. 19-1090.

All of the claims remaining in the application are now clearly allowable. Favorable consideration and a Notice of Allowance are earnestly solicited.

Respectfully submitted,

SEED Intellectual Property Law Group PLLC



E. Russell Tarleton
Registration No. 31,800

ERT:jl

Enclosure:
Postcard

701 Fifth Avenue, Suite 6300
Seattle, Washington 98104-7092
Phone: (206) 622-4900
Fax: (206) 682-6031

717228_1.DOC